ENTREGA FINAL

PREDICCIÓN DE LA SEVERIDAD DE UN ACCIDENTE DE TRÁNSITO EN FUNCIÓN DEL TIPO DE VEHÍCULO

PRESENTADO POR:

LIZETH ANDREA GIRALDO VELEZ
CC: 1001237603

ANGIE LISETH CORONEL YELA
CC: 1085333292

PROFESOR:

RAÚL RAMOS POLLÁN

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

Índice

1. Introducción

Según la OMS los accidentes de tránsito son una de las primeras causas de muerte entre la población de 15 y 29 años de edad a nivel mundial. Por lo que, en el momento que se registra un accidente de tránsito es de vital importancia conocer cuáles son los riesgos potenciales que podrían conducir a un accidente, y a su vez establecer la severidad del mismo. Para este proyecto se plantea implementar un modelo de machine learning con el fin de predecir la severidad de un accidente de tránsito dependiendo del tipo y referencia del vehículo.
Un modelo de machine learning tendrá como salida una variable que se genera a partir del entrenamiento de un algoritmo con datos conocidos. Después del entrenamiento, al aplicar el modelo con una entrada, se espera un pronóstico basado en los datos que entrenaron el modelo. Para este proyecto por la naturaleza de los datos, se implementó un modelo con *Random Forest Classifie*r, y una métrica de clasificación *accuracy* para evaluar el desempeño del modelo.
Como principal resultado, se implementó un modelo de clasificación Random Forest Classifier, el cual se validó con la métrica de clasificación obteniendo una precisión de 0.94 para todo el modelo. Como conclusión general, este modelo podría implementarse en el mercado a la hora de adquirir un vehículo con ciertas características, teniendo en cuenta el grado de severidad del mismo en un accidente al cual puede estar potencialmente asociado.

2. Exploración descriptiva del dataset

El dataset a utilizar fue encontrado en Kaggle y contiene información detallada sobre accidentes viales en Reino Unido, con datos desde 2005 hasta 2014. Este dataset contiene 3 archivos .csv y uno .xls :

1) Un archivo que contiene la información de los accidentes con 32 columnas, llamado "***Accidents0514.csv"***, donde se dan características como condiciones climáticas, severidad del accidente, lugar, fecha, hora, día de la semana, condiciones de la carretera, latitud, longitud, límite de velocidad, etc.

2) Un archivo en el que se presentan las características de los vehículos involucrados en el accidente con 22 columnas llamado " ***Vehicles0514.csv***", que indica el tipo de automóvil, modelo, tamaño del motor, antigüedad del carro, etc.

3) Un archivo donde se recopila la información de las víctimas con 15 columnas llamado "***Casualties0514.csv***", donde se describe la edad, sexo, clase social, etc.

4) Archivo de búsqueda llamado "***Road-Accident-Safety-Data-Guide.xls***": En todos los archivos los datos se encuentran codificados, por lo cual en este se explica la información contenida en los otros 3 archivos.

3. Iteraciones de desarrollo

a) Preprocesado de los datos

El preprocesamiento de los datos es un paso fundamental puesto que los datos reales pueden estar contaminados, que es un caso común, datos incompletos, datos con ruido, datos inconsistentes etc. Esta etapa nos permite generar un conjunto de datos más pequeño que el original, con el fin de mejorar la eficiencia del modelo a implementar [1]. A continuación se enuncian los pasos del preprocesado de los datos implementados.

- Carga de los datos desde la plataforma Kaggle al Google Collaboratory

```
[ ]  !kaggle datasets download -d benoit72/uk-accidents-10-years-history-with-many-variables
```

```
Downloading uk-accidents-10-years-history-with-many-variables.zip to /content
 85% 87.0M/103M [00:00<00:00, 161MB/s]
100% 103M/103M [00:00<00:00, 141MB/s]
```

```
[ ]  !unzip uk-accidents-10-years-history-with-many-variables;
```

```
Archive:  uk-accidents-10-years-history-with-many-variables.zip
  inflating: Accidents0514.csv
  inflating: Casualties0514.csv
  inflating: Road-Accident-Safety-Data-Guide.xls
  inflating: Vehicles0514.csv
```

```
#Cargar los datos
df_accidents = pd.read_csv("Accidents0514.csv")
df_accidents = df_accidents.copy()

df_vehicles = pd.read_csv("Vehicles0514.csv")
df_vehicles = df_vehicles.copy()

df_victims = pd.read_csv("Casualties0514.csv")
df_victims = df_victims.copy()
```

- Identificación de las variables y sustitución de los datos faltantes de los 3 Dataset de estudio (el número corresponde a la cantidad de datos que faltan en cada caso)

```
Accident_Index: 0
Location_Easting_OSGR: 0
Location_Northing_OSGR: 0
Longitude: 1
Latitude: 0
Police_Force: 0
Accident_Severity: 0
Number_of_Vehicles: 0
Number_of_Casualties: 0
Date: 0
Day_of_Week: 0
Time: 0
Local_Authority_(District): 0
Local_Authority_(Highway): 0
1st_Road_Class: 0
1st_Road_Number: 2
Road_Type: 0
Speed_limit: 0
Junction_Detail: 18
Junction_Control: 585322
2nd_Road_Class: 676268
2nd_Road_Number: 16118
Pedestrian_Crossing-Human_Control: 21
Pedestrian_Crossing-Physical_Facilities: 37
Light_Conditions: 0
Weather_Conditions: 161
Road_Surface_Conditions: 2155
Special_Conditions_at_Site: 18
Carriageway_Hazards: 32
Urban_or_Rural_Area: 0
```

```
Accident_Index: 0
Vehicle_Reference: 0
Vehicle_Type: 473
Towing_and_Articulation: 184
Vehicle_Manoeuvre: 1080
Vehicle_Location-Restricted_Lane: 74
Junction_Location: 9876
Skidding_and_Overturning: 132
Hit_Object_in_Carriageway: 87
Vehicle_Leaving_Carriageway: 127
Hit_Object_off_Carriageway: 54
1st_Point_of_Impact: 565
Was_Vehicle_Left_Hand_Drive?: 23049
Journey_Purpose_of_Driver: 44921
Sex_of_Driver: 33
Age_of_Driver: 330608
Age_Band_of_Driver: 330608
Engine_Capacity_(CC): 788471
Propulsion_Code: 770004
Age_of_Vehicle: 874989
Driver_IMD_Decile: 849921
Driver_Home_Area_Type: 588611
```

```
Accident_Index: 0
Vehicle_Reference: 0
Casualty_Reference: 0
Casualty_Class: 0
Sex_of_Casualty: 630
Age_of_Casualty: 45920
Age_Band_of_Casualty: 45920
Casualty_Severity: 0
Pedestrian_Location: 12
Pedestrian_Movement: 15
Car_Passenger: 404
Bus_or_Coach_Passenger: 38
Pedestrian_Road_Maintenance_Worker: 1439022
Casualty_Type: 0
Casualty_Home_Area_Type: 322715
```

Estos datos faltantes se encuentran asignados como “-1”. De forma que se reemplazan estos valores por valores nulos “Nan”, para posteriormente reemplazarlos por un valor estimado.

- <u>Identificación de los tipos de variables del dataset:</u>

Variables numéricas:

```
['Longitude', '1st_Road_Number', 'Junction_Detail', '2nd_Road_Number']
```

```
['Age_of_Casualty', 'Age_Band_of_Casualty']
```

Variables categóricas:

```
(['Sex_of_Casualty',
  'Pedestrian_Location',
  'Pedestrian_Movement',
  'Car_Passenger',
  'Bus_or_Coach_Passenger',
  'Pedestrian_Road_Maintenance_Worker',
  'Casualty_Home_Area_Type'],
 ['Vehicle_Type',
  'Towing_and_Articulation',
  'Vehicle_Manoeuvre',
  'Vehicle_Location-Restricted_Lane',
  'Junction_Location',
  'Skidding_and_Overturning',
  'Hit_Object_in_Carriageway',
  'Vehicle_Leaving_Carriageway',
  'Hit_Object_off_Carriageway',
  '1st_Point_of_Impact',
  'Was_Vehicle_Left_Hand_Drive?',
  'Journey_Purpose_of_Driver',
  'Sex_of_Driver',
  'Age_of_Driver',
  'Age_Band_of_Driver',
  'Engine_Capacity_(CC)',
  'Propulsion_Code',
  'Age_of_Vehicle',
  'Driver_IMD_Decile',
  'Driver_Home_Area_Type'],
 ['Junction_Control',
  '2nd_Road_Class',
  'Pedestrian_Crossing-Human_Control',
  'Pedestrian_Crossing-Physical_Facilities',
  'Weather_Conditions',
  'Road_Surface_Conditions',
  'Special_Conditions_at_Site',
  'Carriageway_Hazards',
  'Did_Police_Officer_Attend_Scene_of_Accident'])
```

- Sustitución de los valores faltantes

Las variables numéricas fueron reemplazadas por su media con la función mean().

```
for i in num_variables_accidents:
  df_accidents[i] = df_accidents[i].fillna(df_accidents[i].mean())
```

```
for i in num_variables_victims:
  df_victims[i] = df_victims[i].fillna(df_victims[i].mean())
```

Las variables categóricas por su moda con mode().

```
for i in cat_variables_accidents:
  print(f" Mode : {df_accidents[i].mode()}")
  df_accidents[i] = df_accidents[i].fillna(df_accidents[i].mode().iloc[0])
```

```
for i in cat_variables_vehicles:
  print(f" Mode : {df_vehicles[i].mode()}")
  df_vehicles[i] = df_vehicles[i].fillna(df_vehicles[i].mode().iloc[0])
```

```
for i in cat_variables_victims:
  print(f" Mode : {df_victims[i].mode()}")
  df_victims[i] = df_victims[i].fillna(df_victims[i].mode().iloc[0])
```

b) Modelos supervisados

Matriz de confusión

Una matriz de confusión, también conocida como matriz de error, es una tabla resumida que se utiliza para evaluar el rendimiento de un modelo de clasificación. El número de predicciones correctas e incorrectas se resumen con los valores de conteo y se desglosan por cada clase [2].

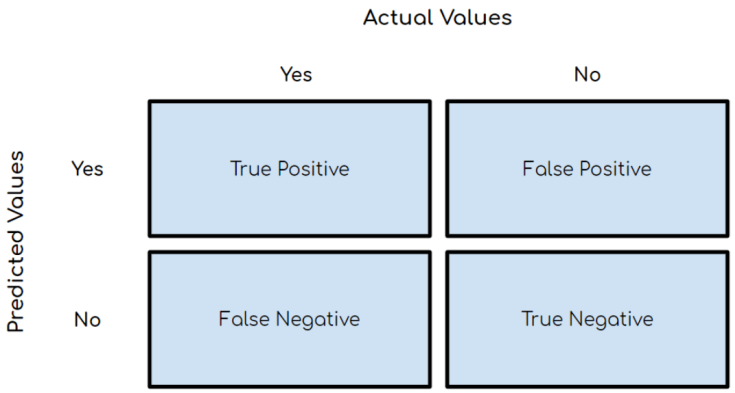


Figura 1. Estructura de una matriz de confusión 2x2

- **Verdadero Positivo (TP):** Resultado en el que el modelo predice correctamente la clase positiva.
- **Verdadero Negativo (TN):** Resultado donde el modelo predice correctamente la clase negativa.
- **Falso Positivo (FP):** También llamado error de tipo 1, resultado donde el modelo predice incorrectamente la clase positiva cuando en realidad es negativa.

- **Falso Negativo (FN):** También llamado error de tipo 2, un resultado en el que el modelo predice incorrectamente la clase negativa cuando en realidad es positiva.

El accuracy o precisión del modelo se mide de forma global y nos da una idea de qué tan frecuentemente el modelo clasifica correctamente. Se calcula como:

$$Accuracy = \frac{TP+TN}{TP+TN+FN}$$

Estos algoritmos se clasifican dependiendo del tipo de variable de salida, suele darse una subcategoría que diferencia entre modelos de clasificación, si la salida es un valor categórico como una enumeración o un conjunto finito de clases. Si la salida es un valor de un espacio continuo se clasifican como modelos de regresión. Para este caso, la variable "Accident Severity" es de naturaleza discreta pero no binaria, por lo que se implementa un modelo de clasificación.

Random Forest

Este es un método versátil de aprendizaje automático que realiza tanto tareas de regresión como de clasificación. Tiene en cuenta métodos de reducción dimensional, trata valores perdidos, valores atípicos y otros pasos esenciales de exploración de datos. Es un tipo de método de aprendizaje por conjuntos, donde un grupo de modelos débiles se combinan para formar un modelo poderoso. Este modelo ejecuta varios algoritmos de árbol de decisiones en lugar de uno solo. Para clasificar un nuevo objeto basado en atributos, cada árbol de decisión da una clasificación y finalmente la decisión con mayor "cantidad de votos" es la predicción del algoritmo [4].

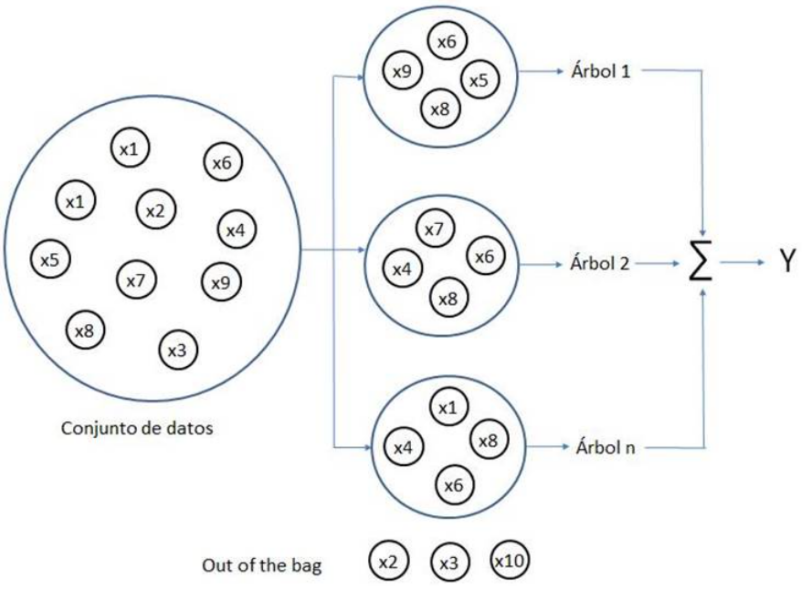


Figura 3. Esquema de un árbol de decisión [5]

Random Forest Classifier de Scikit-learn

Con el fin de mejorar el modelo se aplica un *ensamble method* de Scikit-learn, el objetivo de estos métodos es combinar las predicciones de varios estimadores base construidos con un algoritmo de aprendizaje dado para mejorar la generalización/robustez sobre un solo estimador. Uno de los métodos usados es *averaging method*, su principio fundamental es construir varios estimadores de forma independiente y luego promediar sus predicciones. En promedio, el estimador combinado suele ser mejor que cualquiera de los estimadores de base única porque se reduce su varianza [6]. Para efectos de este proyecto se enuncian tres de sus parámetros ajustados:

- **n_estimators** ***int, default = 100:*** Corresponde al número de árboles
- **max_depth** ***int, default = None:*** Se refiere a la profundidad máxima del árbol
- **class_weight*{"balanced", "balanced_subsample"}, dict or list of dicts, default = None:*** Se refiere a los pesos asociados a las clases. Si no se proporciona, se supone que todas las clases tienen la misma importancia.

Featureimportances() de Scikit-learn

El dataset original puede contener muchas variables que para nuestro modelo no sean relevantes en función de la predicción del mismo, por lo que se entrena el modelo con aquellas variables que son de mayor importancia. En Scikit-learn la importancia de las características las proporciona el atributo ajustado *featureimportances()* y se calculan como la media y la desviación estándar de la acumulación de la disminución de impurezas dentro de cada árbol [7].

c) Resultados, métricas y curvas de aprendizaje

- Generación del Dataset

Se genera un Dataset con los 3 dataset utilizados en el procesamiento de los datos. Para esto se eligió el mínimo valor del total de datos, para estandarizar el tamaño de los dataframe.

```
[28] print(df_accidents.shape)
     print(df_victims_cleaned.shape)
     print(df_vehicles_cleaned.shape)
     # "Accident_Index"

     (1640597, 32)
     (1640597, 15)
     (1640597, 22)
```

Con la función merge() se consolidó un solo data frame, usando "Accident_Index" como *primary key.*

```
df_model_variables = df_accidents.merge(df_victims_cleaned,how="inner",on="Accident_Index")
    .merge(df_vehicles_cleaned,how="inner",on="Accident_Index")
```

- Primer reducción del Dataset

Se eliminan las variables que en un primer análisis se consideraron no aportarían información relevante al modelo.

```
df_model_variables.drop(['Accident_Index','Did_Police_Officer_Attend_Scene_of_Accident','Sex_of_Casualty',
                         'Sex_of_Driver','1st_Road_Class','2nd_Road_Number','Pedestrian_Crossing-Physical_Facilities',
                          'Casualty_Type','Pedestrian_Crossing-Human_Control','1st_Road_Number','Junction_Control',
                         'Bus_or_Coach_Passenger','Day_of_Week','Longitude','Was_Vehicle_Left_Hand_Drive?','Location_Easting_OSGR',
                         'Propulsion_Code','Vehicle_Location-Restricted_Lane','Journey_Purpose_of_Driver','Car_Passenger','Driver_IMD_Decile',
                         'Age_Band_of_Casualty','Age_Band_of_Driver','Latitude','Location_Northing_OSGR','Age_of_Driver','Age_of_Casualty',
                         'Local_Authority_(District)','Police_Force','Pedestrian_Movement','Casualty_Home_Area_Type','Pedestrian_Location',
                         'Casualty_Class','Driver_Home_Area_Type','2nd_Road_Class','Number_of_Vehicles','Light_Conditions',
                         'Vehicle_Leaving_Carriageway','Casualty_Reference','Number_of_Casualties','Pedestrian_Road_Maintenance_Worker',
                         'Local_Authority_(Highway)','Date','Time','LSOA_of_Accident_Location'],axis = 1,inplace=True)
```

- División del dataset en Train - Test

El dataset se divide con un 30% de los datos para validación y un 70% para entrenamiento. Para ello se hace uso de la función ***train_test_split()*** de sklearn, que permite ingresar datos en una sola llamada para dividir y opcionalmente submuestrear los datos.

```
[31] y = df_model_variables['Accident_Severity'].values
     X= df_model_variables.drop('Accident_Severity',axis=1)

     test_size = 0.3
     X_train, X_test, y_train , y_test = train_test_split(X, y, test_size=test_size,random_state=1)
     print(X_train.shape, X_test.shape)

     (1148417, 21) (492180, 21)
```

- Aplicación de los modelos supervisados

Se parte inicialmente definiendo los estimadores correspondientes a los tres modelos a probar. Estos son modelos de regresión, porque aun cuando la variable de salida "Accident_Severity" es de tipo discreta, esta tiene un orden, donde sus valores de salida pueden ser: 1, 2 o 3. De forma que más adelante se explica una forma en la cual se discretiza la variable de salida obtenida, y con base a esto se interpretan los resultados.

```
[32] estimator1 = LinearRegression()
     estimator2 = DecisionTreeRegressor(max_depth=5)
     estimator3 = RandomForestRegressor(n_estimators = 2,max_depth = 5)
```

El mejor estimador se encuentra a partir de una función que implementa el Root Mean Squared Log Error (RMSLE) y una metrica de validacion cruzada ***cross_validate()***, que utiliza diferentes partes de los datos para probar y entrenar un modelo en diferentes iteraciones. Se utiliza principalmente en entornos donde el objetivo es la predicción, y se quiere estimar con qué precisión un modelo predictivo se realizará en la práctica [8].

```
def report_cv_score(z):

    #RMSE en prueba y entrenamiento
    lr_ts_score = z['test_score']*(-1)
    lr_tr_score = z['train_score']*(-1)
    lin_rmse_ts = np.sqrt(lr_ts_score)
    lin_rmse_tr = np.sqrt(lr_tr_score)
    print("RMSLE Test: ", "{:.5f}".format(np.mean(lin_rmse_ts)), "(±", "{:.8f}".format(np.std(lin_rmse_ts)),")")
    print("RMSLE Train: ", "{:.5f}".format(np.mean(lin_rmse_tr)), "(±", "{:.8f}".format(np.std(lin_rmse_tr)),")")
```

```
test_size = 0.3
val_size  = test_size/(1-test_size)
zscores = []
estimators = [estimator1, estimator2, estimator3]
for estimator in estimators:
    print("---------")
    z = cross_validate(estimator, X_train, y_train, return_train_score=True, return_estimator=False,
                       scoring="neg_mean_squared_error", cv=ShuffleSplit(n_splits=10, test_size= val_size,random_state=1))
    report_cv_score(z)
    zscores.append(np.mean(np.sqrt(z['test_score']*(-1))))
best = np.argmin(zscores)
print ("Seleccionado: ", best)
best_estimator = estimators[best]
print ("\n Mejor modelo: ")
print (best_estimator)
```


El mejor resultado se obtiene con el estimador 3, que corresponde al modelo de regresión Random Forest, seguido por el Decision Tree o estimador 2:

```
RMSLE Test:  0.14100 (± 0.00071041 )
RMSLE Train:  0.14064 (± 0.00053239 )
---------
RMSLE Test:  0.14078 (± 0.00066509 )
RMSLE Train:  0.14033 (± 0.00052300 )
---------
RMSLE Test:  0.14072 (± 0.00066014 )
RMSLE Train:  0.14028 (± 0.00053145 )
Seleccionado:  2

 Mejor modelo:
RandomForestRegressor(max_depth=5, n_estimators=2)
```

Ahora es necesario obtener los mejores hiper parámetros de las variables de ambos modelos. Para comenzar, se realiza esta búsqueda con el modelo Random Forest, cuyos parámetros empleados son "***max_depth***" y "***n_stimators***". Para esto se hace uso de ***GridSearchCV*** de sklearn que implementa un método de "ajuste" y "puntuación". Los parámetros del estimador utilizados para aplicar estos métodos se optimizan mediante una búsqueda de cuadrícula con validación cruzada sobre una cuadrícula de parámetros [9]. De la misma manera, se realiza el proceso para el Decision Tree, cuyo parámetro empleado es el "***max_depth***".

```
parametros = {'n_estimators': [2,5,8,12,15],
              'max_depth':[5,7,9]}


model1 = GridSearchCV(estimator = estimator3,
                      param_grid = parametros,
                      cv = ShuffleSplit(n_splits= 5, test_size=val_size,random_state=1),
                      scoring = 'neg_mean_squared_error',
                      verbose = 1,
                      return_train_score = True,
                      n_jobs = -1)

model1.fit(X_train, y_train)
```

```
parametros = {'max_depth': [2,5,8,12,15]}
```

De los códigos anteriores se obtuvieron los mejores hiper parámetros para ambos modelos, arrojando para el Random Forest un "***n_stimators = 15***" y "***max_depth = 9***", de igual manera se obtuvo para el Decision Tree un valor para el parámetro "***max_depth***" de 8.

- Combinación de los modelos supervisados y no supervisados

Posterior al análisis con métodos supervisados, se complementa el estudio con métodos no supervisados antes de comenzar el entrenamiento. En este proyecto se implementa el PCA y NMF para reducir las dimensiones del dataset y conservar los datos de mayor importancia, o sea, aquellos que contienen más información y así entrenar los algoritmos supervisados.

PCA y Random Forest

Se realizó un código que posibilitó explorar varias opciones para el hiper parámetro de este algoritmo con valores de 1, 3, 5, 7 y 9. Se entrenaron los PCA con estos valores y luego se probaron como entrada para el algoritmo Random Forest para saber con cuál se obtenía el mejor resultado, encontrando que con n_components=7 se presentó el mejor desempeño.

PCA

```
components = [1,3,5,7,9]
test_size = 0.3
val_size  = test_size/(1-test_size)
perf = [] #desempeños de los modelos
Rdm_forest = RandomForestRegressor(n_estimators = 15,max_depth = 9)
for i in components:
    pca = PCA(n_components = i)
    X_t = pca.fit_transform(X)

    Rdm_forest.fit(X_train, y_train)
    perf.append(RMSLE(y_test , Rdm_forest.predict(X_test)))
    print('RMSLE del modelo con ', i ,'elementos: ',"{:.5f}".format(RMSLE(y_test , Rdm_forest.predict(X_test))))
    print('------------')

print('Mejor RMSLE: ',"{:.5f}".format(np.min(perf)),'  ; obtenido con ',components[np.argmin(perf)],' componentes para PCA')
```

Para mejorar el modelo se aplica ***RandomForestClassifier()***, teniendo en cuenta los parámetros antes descritos.

```
model_selected = RandomForestClassifier(max_depth=9,n_estimators=15,class_weight="balanced")
model_selected.fit(X_train, y_train)
```

```
RandomForestClassifier(class_weight='balanced', max_depth=9, n_estimators=15)
```

Luego de aplicar el modelo, primero ***class_weigth = "balanced"*** nos permite darle un peso balanceado a las variables, es decir aquellas variables que son menos comunes (en este caso la severidad grado 1 y 2) tendrán mayor importancia durante el entrenamiento del modelo. Adicionalmente, se mencionó que la variable de salida se estudiaría como una variable discreta, sin embargo con este modelo de clasificación se obtuvo una variable de salida de tipo discreta. Con estos resultados se procede a evaluar una métrica de desempeño de clasificación del modelo.

```
y_pred_train = model_selected.predict(X_train)
y_pred_test = model_selected.predict(X_test)
```

```
y_pred_train
```

```
array([3, 3, 3, ..., 3, 3, 3])
```

```
y_pred_test
```

```
array([3, 3, 3, ..., 3, 3, 3])
```

NMF y Decision Tree

El análisis realizado con NMF es igual que con PCA, dado que se buscó el mejor hiper parámetro entre varias opciones, encontrando que con n_components=9. Después de buscar nuevamente los mejores hiper parámetros para el Decision Tree se obtuvieron los mismos antes hallados.

- Aplicación del modelo supervisado

Como la variable de salida "Accident_Severity" puede ser 1, 2 o 3, se requiere el accuracy de los tres posibles casos. Primero se extrajo la matriz de confusión para cada uno de estos. Obteniendo los siguientes resultados:

```
# Grafico de la matriz de confusión multinivel
matrices = metrics.multilabel_confusion_matrix(y_test,y_pred_test)
for i in range(3):
  print(matrices[i])
  graficarMatrizConf(matrices[i],["Negativo", "Positivo"],f"Matriz de confusión para categoria {i+1}",None,False)
```

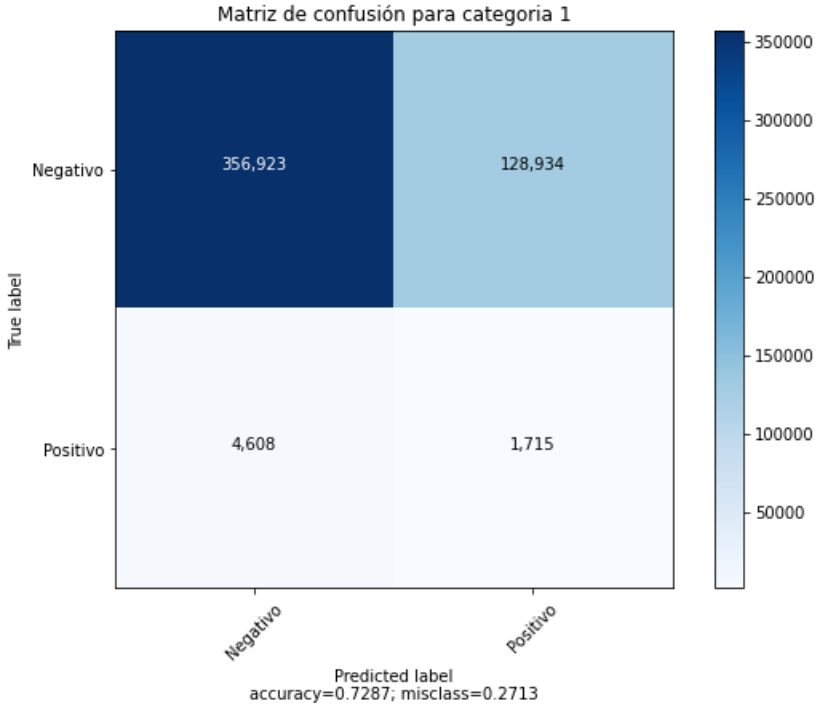


Figura 4. Matriz de confusión para los accidentes de severidad 1

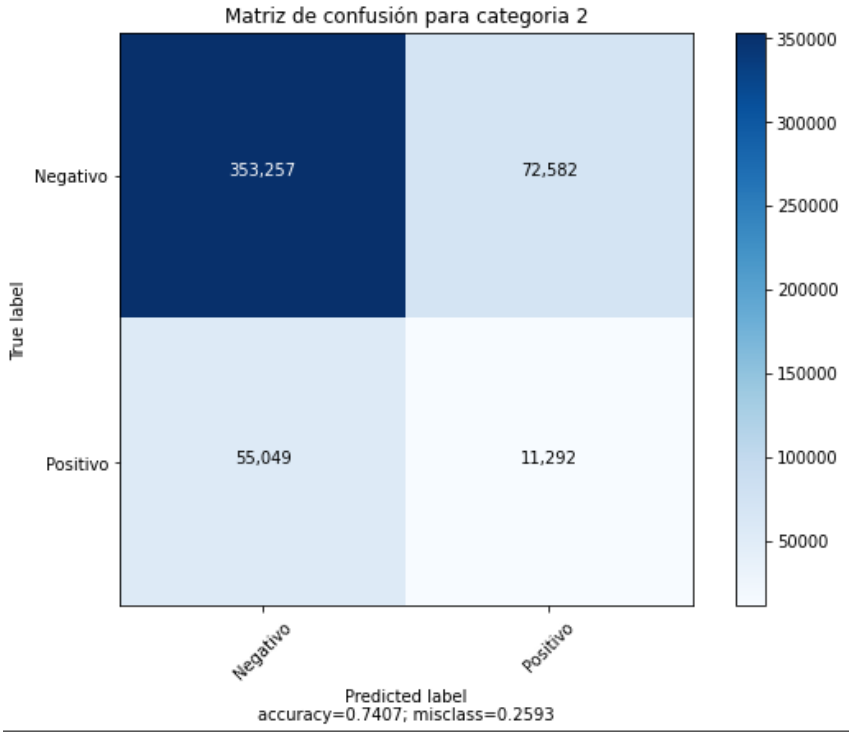


Figura 5. Matriz de confusión para los accidentes de severidad 2

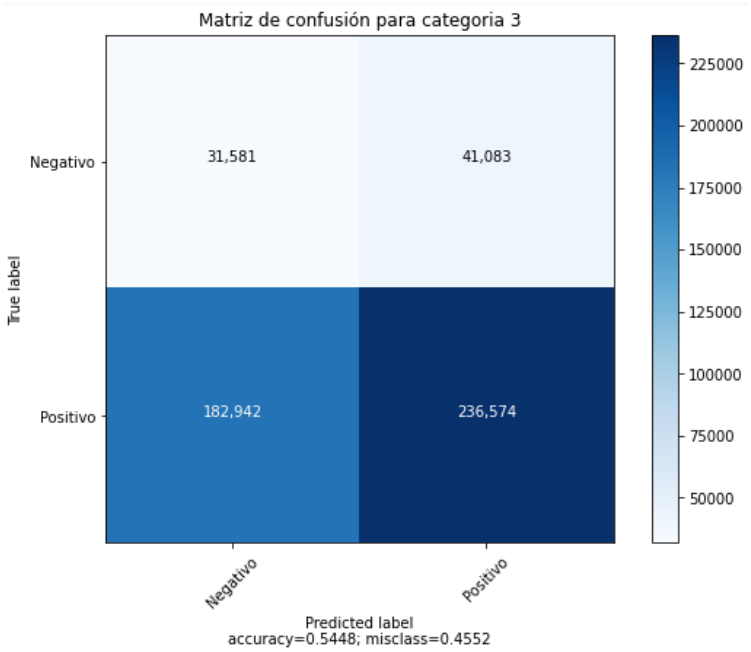
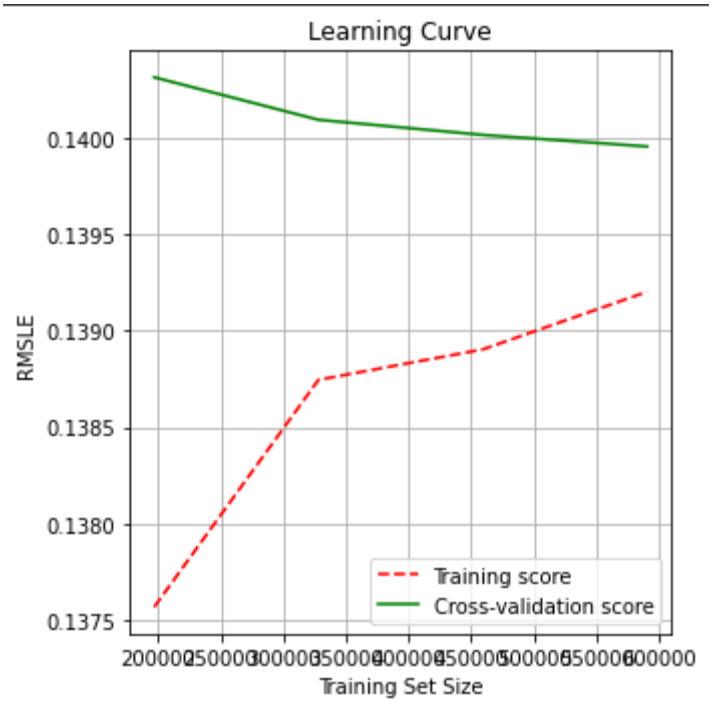


Figura 6. Matriz de confusión para los accidentes de severidad 3

ACCURACY PCA-RANDOM FOREST

Por lo que una vez aplicado el modelo de validación, para una aplicación en el mercado, sobre la evaluación del grado de severidad de un accidente, dependiendo del tipo de vehículo que se tiene, se plantea que con una precisión mayor a 0.94 el modelo desarrollado es útil en el mercado y puede ser un estimador confiable a la hora de adquirir un vehículo evaluando el riesgo de tener un accidente de transito severo, medio o leve, sin embargo la precisión que se obtuvo con este modelo fue de 0.54, de forma que la predicción del modelo no es confiable.

CURVA DE APRENDIZAJE RANDOM FOREST



La curva de aprendizaje no muestra que, la línea roja corresponde a la de entrenamiento y la línea verde al modelo de PCA + Random Forest, de forma que se tiene un RMLE bajo pero dada la brecha que existe entre este valor y el valor de prueba, se tiene un caso de overfitting, lo que puede indicar que los datos no son suficientes o se tiene una hipótesis del modelo muy compleja.

FeatureImportances()

Con el fin de mejorar el desempeño del modelo, se eliminan las variables que pasaron un primer filtro, pero que sklearn permite descartar aquellas variables que no son de importancia en el modelo, y pueden ser eliminadas.

```
importances = model_selected.feature_importances_
feature_names = X.columns
std = np.std([tree.feature_importances_ for tree in model_selected.estimators_], axis=0)
forest_importances = pd.Series(importances, index=feature_names)
fig, ax = plt.subplots(figsize=(12,6))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```

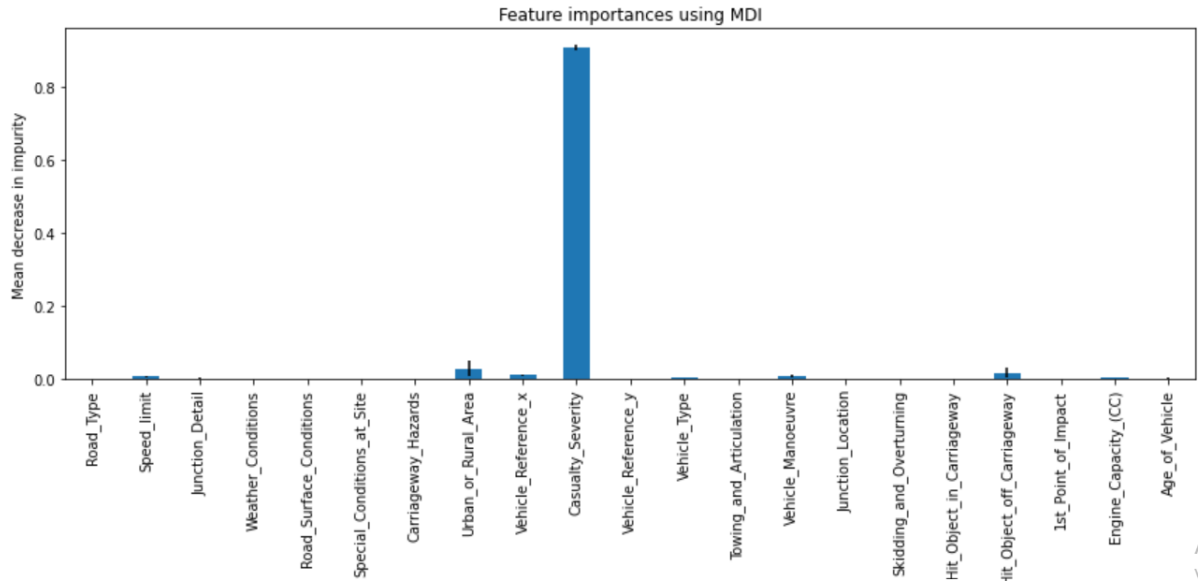


Figura 8. Feature importances empleando MDI

Según los resultados, las variables con mayor importancia en el desempeño del modelo son: ***Speed_limit, Urban_or_Rural_Area, Vehicle_Reference_x, Casualty_Severity, Vehicle_Reference_y, Vehicle_Type, Vehicle_Manoeuvre, Hit_Object_off_Carriageway, Engine_Capacity.*** En un primer análisis podemos decir que en función de predecir la severidad de un accidente dependiendo del tipo de vehículo, las variables mencionadas, se relacionan con las características que podrían diferenciar un tipo de vehículo de otro. De forma que los resultados obtenidos, son acordes a lo esperado. De manera que si se quisiera mejorar el modelo planteado, sería necesaria la implementación de la información recopilada

en el *featureimportances,* para descartar aquellas variables que no son necesarias para la predicción.

4. Retos y consideraciones de despliegue

Manejar los diferentes tipos de variables presentes en nuestro dataset representó para nosotras un verdadero reto, dado que se comenzó trabajando la variable objetivo pensando que era tipo continua ya que es ordenada, pero se empleó un modelo que permitió discretizarla, el Random Forest Classifier.

Por otra parte, para desplegar el modelo en producción en Colombia se hace necesaria la capacitación del personal de tránsito, autoridad nacional autorizada para atender los accidentes viales, con el fin de que sea esta entidad competente la encargada de nutrir la base de datos después de cada accidente, teniendo en cuenta las características particulares de los vehículos nacionales, dado que este modelo fue empleado con datos de accidentes viales en el Reino Unido, razón por la cual la severidad de los accidentes se vería afectada si no se actualiza la base de datos con los vehículos de territorio nacional. En consecuencia, la utilización de una base de datos es requerida para actualizar la información diariamente con los accidentes viales reportados, garantizando además, la actualización de datos que permita el adecuado funcionamiento del modelo.

5. Conclusiones

- A partir de los resultados obtenidos se puede concluir, que el modelo propuesto no proporciona una medida confiable al momento de predecir el grado de severidad de un accidente asociado a las características de un vehículo, como tipo, referencia, límite de velocidad, capacidad del motor, entre otras, pero puede mejorarse a partir del número de datos o disminuyendo la complejidad del modelo.
- Mediante la realización de este proyecto se comprueba que para la implementación de un modelo es importante probar con diferentes algoritmos y a su vez, con diferentes parámetros, con el fin de analizar cuál proporciona un mejor desempeño y tratamiento de los datos.
- El atributo ***Featureimportances*** de Scikit-learn es una herramienta bastante potente dado que permite conocer cuáles son las variables más importantes empleadas en el funcionamiento del modelo.
- Se evidenció la importancia de estudiar de manera detallada el dataset a trabajar, dado que en algunos casos, como se presentó en este proyecto, es necesario realizar un balance y darle mayor peso a las variables menos comunes para el entrenamiento del modelo y así permitir el correcto funcionamiento del mismo.
- El éxito de un modelo depende en gran medida del preprocesamiento de los datos que se usan para el entrenamiento, esto dado que si no se realiza un correcto tratamiento de los datos Nan, por ejemplo, la precisión y predicción del modelo se verá afectada.
- Una métrica de validación del modelo de gran utilidad es el accuracy para variables discretas.

6. Referencias bibliográficas

[1] Gonzalez, L. (2022, 2 septiembre). *Aprendizaje Supervisado: Random Forest Classification*. 🤖 Aprende IA. Disponible en: https://aprendeia.com/aprendizaje-supervisado-random-forest-classification/

[2] Shin, T. (2020, 15 mayo). *Comprensión de la Matriz de Confusión y Cómo Implementarla en Python*. DataSource.ai. Disponible en: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

[3] *Curva ROC*.(2022, 29 agosto). Disponible en: https://es.wikipedia.org/wiki/Curva_ROC

[4] Herrera, F., Riquelme, J. & Ruiz, R. (s. f.). Preprocesamiento de datos [Diapositivas; Digital]. Disponible en: http://www.lsi.us.es/redmidas/IIreunion/trans/prepro.pdf

[5] Espinoza, J. (2020, septiembre). *Application of random forest and XGBoost algorithms based on a credit card applications database*. Scielo. Disponible en: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432020000300002#:~:text=El%20algoritmo%20Random%20Forest%20

[6] 1.11. Ensemble methods. (s. f.). scikit-learn. Disponible en: https://scikit-learn.org/stable/modules/ensemble.html

[7] *Feature importances with a forest of trees*. (s. f.). scikit-learn. Disponible en: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

[8] *Cross-validation (statistics)* (2022, 5 noviembre) Disponible en: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[9] *sklearn.grid_search.GridSearchCV — scikit-learn 0.17.1 documentation*. (s. f.). https://scikit-learn.org/0.17/modules/generated/sklearn.grid_search.GridSearchCV.html

[10] *UK Accidents 10 years history with many variables.* Disponible en: https://www.kaggle.com/datasets/benoit72/uk-accidents-10-years-history-with-many-variables?select=Accidents0514.csv